

Learning Packet Analysis with Data Science



Ronald Eddings

Follow

Jul 31, 2018 · 5 min read

Have you ever opened Wireshark and thought, “this is nice, but sometimes filtering and following TCP streams is tedious?” If not, open Wireshark more. In this post, I’ll cover how to leverage Python, Scapy, Pandas, and Seaborn to bring excitement back to packet analysis. Also, this post will serve as a prequel for our next series on Packet Manipulation with Data Science.

No.	Time	Source	Destination	Protocol	Length	Info
19	11.977411	192.168.1.2	192.168.1.157	TCP	60	60 → 54419 [ACK] Seq=54419 Win=0 Len=0
20	11.977416	192.168.1.157	192.168.1.2	TCP	66	66 → 54419 [FIN, ACK] Seq=1 Ack=2 Win=5792 Len=0 TSval=18540
21	13.674543	Vmware_b0:8d:62	Vmware_b0:8d:62	ARP	60	Who has 192.168.1.30? Tell 192.168.1.10
22	13.674786	Vmware_b0:8d:62	Vmware_b0:8d:62	ARP	60	192.168.1.30 is at 00:0c:29:69:e6:2b
23	18.878898	192.168.1.158	64.12.24.50	SSL	60	Continuation Data
24	18.871477	64.12.24.50	192.168.1.158	TCP	60	443 → 51128 [ACK] Seq=1 Ack=7 Win=64240 Len=0
25	33.914966	192.168.1.158	64.12.24.50	SSL	243	Continuation Data
26	33.915486	64.12.24.50	192.168.1.158	TCP	60	443 → 51128 [ACK] Seq=1 Ack=196 Win=64240 Len=0
27	34.806599	192.168.1.158	64.12.24.50	SSL	94	Continuation Data
28	34.806684	64.12.24.50	192.168.1.158	TCP	60	443 → 51128 [ACK] Seq=1 Ack=236 Win=64240 Len=0

Is it just me or does Wireshark remind you of a Pandas DataFrame?

Should I be learning Python and Data Science?

The short answer is, Yes. We’re living in a time where both the cybersecurity analyst and researcher are losing their technical edge by relying on far too many closed-source tools. Combining Python and Data Science enables anyone to reach new realms of possibilities in cybersecurity. Not only will it give you the ability to manipulate data, you can begin creating custom visualizations with the data that you’ve manipulated.

This Post will Empower You To:

- Capture Packets with Scapy
- Manipulate Data with DataFrames
- Create visualizations that describe packet capture

- Glean out suspicious conversations on the packet level

Where Do We Begin?

If you haven't gotten around to using Jupyter Notebook, please start today—This should serve as your playground for executing and bringing ideas to life. We'll also leverage Anaconda as our package manager to make our lives simple while creating our development environment:

Steps:

1. Download and install Anaconda (Python 3.6) [here](#)
2. Navigate to Anaconda Cloud and download the Packet-Analytics Project [Here](#):
3. In a terminal window, run the following anaconda command:

```
anaconda-project unarchive packet-analytics.tar.bz2
```
4. In the same terminal window, switch to the packet-analytics directory:

```
cd packet-analytics
```
5. In a terminal window, run the following anaconda command:

```
anaconda-project run
```

 (Note you may need to sudo since we'll be sniffing packets)

If you're interested in following along with a Docker image, follow this [link](#):

Let's Get Started

There's a few libraries that we'll need to import to make all of this possible. Take a look at the gist below:

Scapy vs Everything

Scapy is a powerful tool that will give us the ability to capture, load, and save packets. Scapy is my packet capture tool of choice because of its extensibility and ongoing community support. Typically, packet capture and/or analysis is performed in Wireshark. It's tough to keep track of multiple suspicious indicators in Wireshark while also keeping track of multiple connections. I've also witnessed attempts and efforts

to store packet captures in Splunk and other Search Engines—This is not a cheap solution and does not solve the problem of analysis. When manipulating packets in Scapy, operations can seem a bit rigid. Transforming our packet capture into a Pandas DataFrame will make our lives a lot easier.

Sniffing Packets In Scapy

Sniffing packets in scapy is a straightforward process. We have the option to define a finite numbers of packets to sniff.

Appending a PCAP

For the sake of this post, I've created a PCAP with a seemingly suspicious stream. We'll append the suspicious PCAP to what we've already collected

504 Gateway Time-out

nginx


Exploring An Item In Packet List

When enumerating through the collected PCAP (PacketList Object), we can observe encapsulation of network layers. More specifically, pcap[101]'s first layer is Ethernet -> which has a payload (IP) -> Which has a payload (UDP). It's important to understand encapsulation because data can be communicated via tunneling protocol (MPLS, GRE, ect..)

Converting PCAP to Pandas DataFrame

If you've searched "Python Data Science" on Google then you've surely come across Pandas. This is an astounding tool to leverage when attempting to manipulate various data types. From Pandas, we'll be leveraging DataFrames—two dimensional data structures. We'll convert boring Scapy output to a more comprehensive DataFrame:

###[Ethernet]###	
dst = 88:e9:fe:6a:92:52	
src = 80:37:73:96:9b:db	
type = 0x800	
###[IP]###	
version = 4	
ihl = 5	
tos = 0x20	
len = 84	
id = 58919	
flags =	
frag = 0	
ttl = 122	
proto = udp	
chksum = 0x360c	
src = 84.54.22.33	
dst = 10.1.10.53	
\options \	
###[UDP]###	
sport = domain	
dport = domain	
len = 64	
chksum = 0xfe25	
###[DNS]###	
id = 12	
qr = 1	
opcode = QUERY	
aa = 0	
tc = 0	
rd = 1	
ra = 1	



src	dst	sport	dport	payload
10.0.0.175	52.89.46.250	63928	443	65
52.89.46.250	10.0.0.175	443	63928	0
10.0.0.181	224.0.0.251	5353	5353	78
10.0.0.38	224.0.0.251	5353	5353	906
94.130.23.236	10.0.0.175	443	64262	1448
10.0.0.175	94.130.23.236	64262	443	0
94.130.23.236	10.0.0.175	443	64262	1448
94.130.23.236	10.0.0.175	443	64262	1448
10.0.0.175	94.130.23.236	64262	443	0
10.0.0.175	94.130.23.236	64262	443	0
94.130.23.236	10.0.0.175	443	64262	1448
94.130.23.236	10.0.0.175	443	64262	1448
10.0.0.175	94.130.23.236	64262	443	0
10.0.0.175	94.130.23.236	64262	443	0
10.0.0.175	54.241.162.64	61970	443	154
94.130.23.236	10.0.0.175	443	64262	1448
94.130.23.236	10.0.0.175	443	64262	1448

The below code snippet is important but can be skipped. The key takeaway of the code snippet is that we're enumerating through the encapsulated network layers and creating a DataFrame that will have encapsulated layer fields as columns (ie. Source Address, Source Port, ect...)

DataFrame Basics

There are many amazing DataFrame introduction posts that can be found here and here. The below snippet has some DataFrame Basics to keep in your back pocket:

Working with DataFrames is easier than you think! 😊

PCAP Statistics

When dealing with suspicious, you don't want to spend countless hours on determining if something is malicious or benign. We can leverage our DataFrame and gather quite a bit of statistics on what is encompassed in our PCAP. In this example, we're going to retrieve the most frequent source address, destination address and some information about the ports that are being utilized in network communications:

```
# Top Source Address
count          118
unique         10
top            10.1.10.53
freq           31
Name: src, dtype: object

# Top Destination Address
count          118
unique          9
top            10.1.10.53
freq           31
Name: dst, dtype: object

# Who is Top Address Speaking to?
['84.54.22.33' '75.75.75.75']

# Who is the top address speaking to (Destination Ports)
[53]

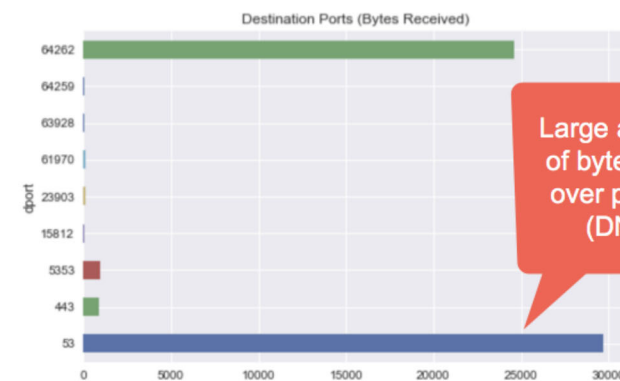
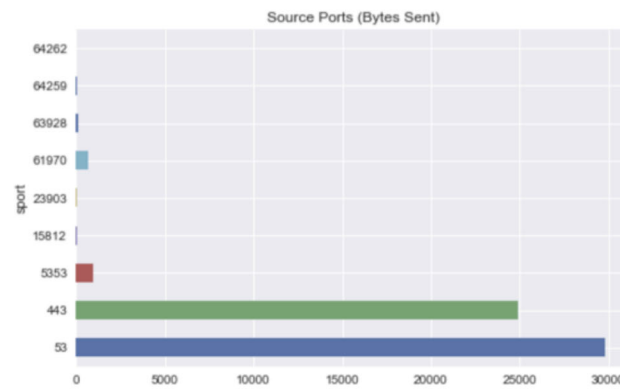
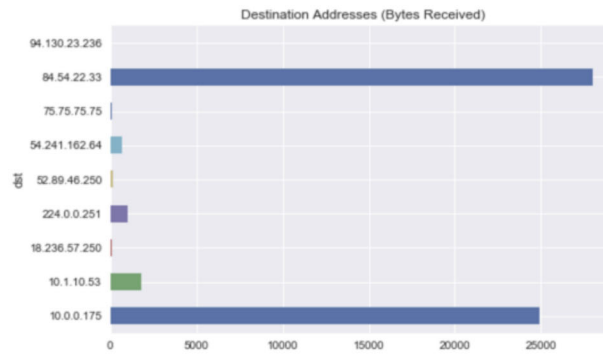
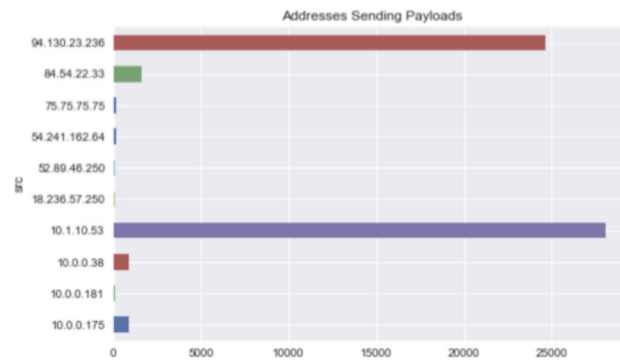
# Who is the top address speaking to (Source Ports)
[53 15812 23903]
```

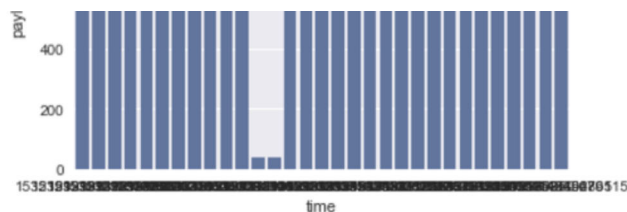
Visualizations

When dealing with large quantities of packets, creating visualizations in the form of graphs can be quite helpful for understanding trends and patterns. We can leverage matplotlib, seaborn, or other libraries such as plotly to visualize our data. When searching for malicious artifacts in PCAP graphs can highlight anomalous spikes of data and display that data is being sent out in regular intervals (or pseudorandom intervals). In the below snippet, we begin applying grouping operations—Grouping data by Destination Address and other attributes:

504 Gateway Time-out

nginx





Payload Investigation

The graphs that we created highlighted the fact that a large amount of data was sent over port 53. Exfiltrating data using this port is a common technique for attackers due to the fact that restricting DNS communication can be troublesome. At this point, we can open Wireshark or **write a few lines of code to make this action repeatable**. We'll perform another grouping operation, separate the conversation into its own dataframe, and view the suspicious conversation:



Conclusion

Working with PCAP can sometimes be tedious and intimidating. By leveraging Python and Data Science techniques we can begin to glean out more interesting attributes and find the suspicious or malicious data that we're searching for quickly. By using these techniques we're only scratching the surface of possibilities and with practice you'll begin to find many artifacts in your packet streams. My challenge to the reader is to fully extract the PNG from the PCAP and display the image.

...

Thanks For Reading. If you enjoyed this post please
give it 50 claps (Yes, 50 claps 😊👏👏👏👏)

- Join our Slack channel
- Follow us on YouTube, Twitter, Reddit, Facebook

